Tools for the LGR Creation, Management and Use

Version 0.9

1. Introduction

ICANN hosts a repository of language and script specific data through its IANA department which is used by registries to generate labels, generally known as "tables" and now referred to as Label Generation Rulesets (LGRs). A new machine-readable XML-based format is currently being designed to organize and represent LGR data. This format will also be used for the LGR for the Root Zone, for validating prospective top-level domain (TLD) labels and determining the variants of existing and prospective labels. The Root Zone LGR is currently being developed with the collaboration of various script communities (organized as script based Generation Panels).

The formal representation is captured in "Representing Label Generation Rulesets using XML (draft-davies-idntables-08)", which is work in progress and available at https://tools.ietf.org/html/draft-davies-idntables (the XML Format). This specification may be revised in the future, but it is assumed to be mostly stable and only minor edits are likely to occur in the future prior to finalization.

This document describes the potential users and use cases for the LGR and details the relevant requirements for development of LGR Tool Set.

2. LGR Users and Use Cases

There are at least three kinds of high-level use cases, with multiple users, for this technology.

2.1. Use Case 1: Generation Panels and Registries – LGR Creation

As a first step, before an LGR can be used, the LGR must be created. The LGR may be created for the top-level or for second and other levels. For the top-level, the LGR is being created by a process involving communities organized into Generation Panels (GPs), who have the necessary expertise to work on the content of the LGR but may not have necessary expertise to author the LGR file in the XML Format. Thus, an easy to use interface is needed for the GPs to input linguistic data for an LGR and create it in the XML Format.

Further, as the use of the XML Format is adopted by the DNS industry, other stakeholders, including domain registries at the second and other levels, will also benefit from tools to convert their data into the XML Format.

2.2. Use Case 2: TLD applicants and End-Users – LGR Use

Once an LGR is created, it is intended to be used to validate if a label is allowed per the rules being captured by the LGR. For example, TLD applicants may use a tool supporting the XML Format to use the Root LGR to determine (i) if the label being applied for meets the requirements of a TLD label as allowed by the IDNA2008 standard and constrained further by the community through the LGR for the Root Zone, and (ii) what are the variant labels (if any) of the label, along with dispositions of these variants (such as allocatable or blocked). End-users may use the same tool to determine the validity of a label for second or other levels, if the LGR is made available by the relevant registry in the XML Format.

2.3. Use Case 3: Integration Panel and Registries – LGR Management

As the LGR is being created and maintained, there may be additional management operations needed. This includes merging two LGRs into a single combined one, comparing two LGRs to determine similarity (or differences) between them, and other such functions. For example, the Integration Panel tasked to finalize the LGR for the Root Zone may need to merge LGR proposals by different GPs together or compare and highlight the differences between two different GP proposals which use an overlapping subset of code points. Similarly, registries may require such functions to add support for more languages or scripts over time. Another management function is for the tool to be able to convert the language tables from (some of) the existing formats into the XML Format.

3. Scope of Work

This current work aims to address the above mentioned three use cases. This will be done in three separate phases.

Phase 1 of the development will focus on LGR creation and will aim to develop a web-based application which allows for users to construct a LGR through an interactive interface, and have the ruleset export into a syntactically-valid LGR in the XML Format. The minimum level of functionally will provide an interface to define code point eligibility and variant rules. Support for any label level restrictions using Whole Label Evaluation (WLE) rules is optional in Phase 1, as the number of uses for WLE rules is expected to be minor and can be hand-written in the XML format. In later development phases, a WLE rule development interface may also be added, based on community need.

In Phase 2, the development will focus on using the LGR, and will allow users to select a pre-defined LGR in the XML Format, and then either validate a label or generate its variant labels.

In Phase 3, multiple management functions will be developed, including the conversion of language tables into the XML Format, comparing two LGRs, and additional operations including union, intersection and difference of two LGRs.

The software will be designed to have a library of functions which can be re-used across the three use cases. These functions will not only be used by the interface but may also be included by a third-party software.

4. Functional Requirements

This sections gives detailed requirements for the three phases.

4.1. Phase 1: LGR Creation - Functional Requirements

1. The system will conform to the specifications, input choices and output format given in the XML Format, even if not completely specified explicitly in the requirements below.
2. The system will allow multiple input methods for users, as per the details given below. The system shall not permit input of code points disallowed or unassigned in IDNA2008. The characters allowed for input should be based on the script(s) selected by the user and on the version of IDNA2008 tables (and corresponding Unicode data file) specified by the user. The software should have the

ability to have the Unicode data updated to newer versions of Unicode easily by updating the requisite data files in the software package.

2.1. Input code point by keying in a Unicode character, e.g. "ب"
2.2. Input code point by keying in the hexadecimal value of the Unicode code point e.g. "0628" (for ب)
2.3. Input a range of code points by one of the two methods in 2.1 or 2.2, e.g. "a .. z", or "0061 .. 007A"
2.4. Select (and de-select) one or more code points from the list of code points (both Unicode value and character displayed in a linear list).
2.5. Input a text file in one of the three plain text formats: (i) simple one rule per line (to be defined), (ii) the format described n RFC 3743, and (iii) the format described in RFC 4290. When input is from a file, the user may select to process it in a "manual" mode or an "automatic" mode.
    2.5.1. The manual mode will require the system to prompt input for each line and allow user to accept or skip that line for incorporating in the LGR. All lines skipped must be logged in a log file that can be used for debugging, including time stamps and the text content skipped.
    2.5.2. In the automatic mode, the system will process the entire file without user interaction, logging any erroneous lines or which it is unable to process and incorporate.

3. The system should allow user to input metadata as specified in the XML Format, including the elements specified in 3.1 – 3.8 below. For each element, where available, the system should allow look-up for the user to select among available options, showing both human readable and formal values of these elements. For compound elements, the system should accept input of each portion separately and compound them internally. Clear and precise instructions should be displayed on the screen during the process for users to select the appropriate input.
    3.1. The version element – numerical positive value (e.g. 1, 2, 3, etc.) and associated comment
    3.2. The date element using a date-picker – internally stored in a valid ISO 8601 date string as described in RFC 3339 – using current system date as default value.
    3.3. The language element - must be a valid language tag as described in RFC 5646 – the system will take input for each field (language, script, etc.) separately from users using a look-up field, where possible– and create the complete tag internally. Commonly used English names should also be displayed in addition to the formal names to facilitate user input. Language element can be repeated to allow for adding multiple language elements.
    3.4. The scope element – optional element which may contain a domain name to which the policy is applied.
    3.5. The description element - optional free-form element – has a "type" attribute which should be a valid MIME type, and refers to the media type of the data, with typical types as "text/plain" or "text/html". "text/plain" will be assumed if no type attribute is specified.
    3.6. The validity-start and validity-end elements – optionally specifying the start and end times as an RFC 3339 date-time string (e.g. "2014-11-12T04:28:00Z") during which the LGR is valid. There is no default specified.

3.7. The Unicode-version element – look up from stable Unicode versions – would need to be updated as stable Unicode versions are released.
3.8. The references element – optional as per details in section 4.3.8 in http://tools.ietf.org/html/draft-davies-idntables.

4. User must select a starting code point repertoire. User should be given the following choices by default, but may select another alternate reference file. All these files must be in the XML Format.
   4.1. IDNA2008 PVALID and CONTEXT O/J characters based on Unicode 6.3.0 – system should not allow Unicode code points which are DISALLOWED by IDNA 2008. The default should be configurable to include later updates to IDNA 2008 repertoire, based on Unicode releases, from the IANA pages, e.g. http://www.iana.org/assignments/idna-tables-6.3.0/idna-tables-6.3.0.xhtml for Unicode version 6.3.0.
   4.2. Maximal Starting Repertoire (MSR-2). The default should be configurable to include later releases of the MSR.
   4.3. The system should check each discrete permissible code point being added in the LGR by the user against the selected reference list and give warning to the user if a code point does not exist in the reference. The user may over-ride the reference and may still include the code point. All such code points, which are over-ridden must be logged. However, the user must not be allowed to override the IDNA2008 restrictions.
5. User may also define a sequence of two or more code points in a LGR – for example, when defining the source for n:m variant mappings.
6. The system should allow users to enter one or more variants of (a) a code point, or (b) a code point sequence. Each variant may be a single code point or a sequence of code points.
7. The system should check for symmetric (if A is a variant of B, then B is a variant of A) and transitive (if A is a variant of B and B is a variant C then A is also a variant of C) relationship between variants on the request for user or before finalizing the LGR.
8. The system should also allow users to select the type for each code point variant. The type for each code point variant should be “block” by default.
9. The system should allow for defining conditional code point variants. The user should select the “when” or “not-when” relationship and the set of applicable elements or label to represent such a set.
10. Uniqueness of code point variants against a code point must be determined and any input should be rejected if it introduces repeated entries. It should be noted that the same variant can be repeated if its “when” or “not-when” attributes are different.
11. Any "char", "range" or "variant" element may contain a comment in a "comment" attribute. The contents of a comment attribute are free-form plain text. If the system breaks a range into individual elements internally, the comment must be duplicated for each such entry created.
12. The system should also allow users to add reference to each code point or code point sequence. References must be defined before being looked up. When looking up, the user should see the reference no. and reference details, though only the former is stored internally when selected. User should also have “add new reference” option, in case a new reference is needed at the time of association.
13. The system should allow users to create, view and edit incomplete versions of LGR, in the XML Format. See Requirement 2.4 in Phase 1 as a display option to view data. Data may be added by the options given in Requirement 2 of Phase 1.

13.1. Allow user to create a new LGR.
13.2. Allow user to save intermediate (incomplete) version of an LGR being created. The system should check and log the required elements not present in the intermediate version while saving it.
13.3. Allow user to load for further alteration an intermediate (incomplete) version of a LGR already saved.

14. Starting from a new or an intermediate (incomplete) version of a LGR already saved, the system should allow the user to finalize for release and export a LGR. All necessary checks for completion and correction must be performed at this time and release should only be possible if the criteria specified in the XML Format are met.
15. All text labels, look up choices, messages and error messages should be in English, but the interfaces should be internationalized, loading text from a resource file, so that they may be localized to other languages. By selecting appropriate languages (English by default) the system should be able to display interface in other languages for which the resource files have been translated.
16. The system should generate the summary of the LGR built, including counts of code points, variants by type, count of rules, counts by script code, and number of code points sharing a tag value.

4.2. Phase 2: LGR Use - Functional Requirements

The system should provide interface using a web browser or and API for the following functionalities.

1. The system should associate an LGR in the XML Format from one of the following categories:
   1.1. Default LGRs (built into the system):
      1.1.1.IDNA2008 for the Unicode version available at http://www.iana.org/assignments/idna-tables/
      1.1.2.MSR-2
      1.1.3.LGR-1
   1.2. User defined LGR
2. The system should input a (i) U-label, or (ii) an A-label, or (iii) Unicode code point sequence and tell if the label is valid as per the reference LGR associated. The system should output the label in U-label, A-label and Unicode code point sequence format along with the validity information.
3. The system should list all the variants of the label along with their dispositions. The variant labels should be displayed in U-label, A-label and Unicode code point sequence formats. Note that the list of variants can grow very large and for the web interface the software should be designed to handle calculations and user-friendly display of such large sets.
4. If the label is valid, the system should give details of the rules applied for the label (if the user asks for details) and if the label is not valid, the system should in addition list the portions of label and the reasons for which the label has failed.
5. For the list of variant labels, the system should output the rules used to generate them (if the user asks for the details).

4.3. Phase 3: LGR Management - Functional Requirements

1. A code point may also be given a tag attribute.

2. The system should allow for users to define classes based on (a) tags, (b) Unicode property, (c) user-defined sets, and (d) combined classes, which can be created by (i) Union, (ii) Intersection, (iii) Difference, and (iv) Symmetric difference of existing classes.
3. The system should allow input of WLE rules using the tags and rules defined.
4. The system should be able to input two LGRs (in the XML Format) and compare and output the similarities and differences.
5. The system should be able to input two LGRs (in the XML Format) and output their intersection.
6. The system should be able to input two LGRs (in the XML Format) and output their union.
7. The system should be able to completely specify the LGR by converting ranges into code point level entries.

5. Non-Functional Requirements

The following non-functional requirements are applicable to all the phases of the development of the LGR Tool Set.

1. The system should be able to handle large enough amounts of data to develop a single LGR– which may contain a significant part of the Unicode standard.
2. The system should be developed using the Python programming language, which can be licensed with open source license. The system must not contain any third party code which prevents it from release with licensing requirements stipulated.
3. The system should have an extensible architecture to include other possible processing functions. Further, the logic of the web interface and the underlying LGR manipulation should be sufficiently separated such that alternative user interfaces may be developed using the same LGR manipulation library functions.
4. The system code must be properly commented to ensure facilitating internal or third party extension for maintenance by ICANN or the community.
5. The system interface should be supported by commonly used web browsers, including Internet Explorer, Chrome, Firefox and Safari.
6. Multiple users must be able to access this application at the same time, without concurrency issues, using web and other interfaces provided.
7. System should allow the user to choose a font available in it for display and should not require special fonts. It is assumed that some limitations of display may exist based on the font selected from the available options.
8. All data for the system must be stored in text files.